5.1传输层提供的服务

- 传输层的功能
  - 概述
    - 传输层向它上面的应用层提供通信服务，它属于面向通信部分 的最高层，同时也是用户功能中的最低层
    - 为运行在不同主机上的进程之间提供了逻辑通信
  - 传输层的功能
    - 输层提供应用进程之间的逻辑通信（即端到端的通信）
    - 复用和分用
      - 复用：发送方不同的应用进程都可使用同一个传输层协议传送数据
      - 分用：接收方的传输层在剥去报文的首部后能够把这些数据正确交付到目的应用进程
    - 传输层还要对收到的报文进行差错检测（首部和数据部分）
    - 提供两种不同的传输协议，即面向连接的TCP和无连接的UDP
    - 向高层用户屏蔽低层网络核心的细节（如网络拓扑、路由协议等），使应用进程好像在两个传输层实体之间有一条端到端的逻辑通信信道
- 传输层的寻址与端口
  - 端口的作用
    - 端口可以标识主机中的应用进程
    - 让应用层的各种应用进程将其数据通过端口向下交付给传输层，以及让传输层知道应当将其报文段中的数据向上通过端口交付给应用层相应的进程
    - 端口号只有本地意义，在因特网中不同计算机的相同端口是没有联系的
  - 软件端口与硬件端口
    - 软件端口：协议栈层间的抽象的协议端口，是应用层的各种协议进程与传输 实体进行层间交互的一种地址
    - 硬件端口：不同硬件设备进行交互的接口
    - ***传输层使用的是软件端口***
  - 端口号
    - 服务端使用的端口号
      - 熟知端口号（0∽1023）
        - FTP 21
        - TELNET 23
        - SMTP 25
        - DNS 53
        - TFTP 69
        - HTTP 80
        - SNMP 161
      - 登记端口号（1024∽49151） 使用这类端口号必须在IANA登记，以防止重复
    - 客户端使用的端口号（49152∽65535）
      - 又称短暂端口号（也称临时端口）：这类端口号仅在客户进程运行时才动态地选择
      - 通信结束后，刚用过的客户端口号就 不复存在，从而这个端口号就可供其他客户进程使用
  - 套接字
    - 背景
      - IP地址来标识和区别不同的主机
      - 端口号来标识和区分一台主机中的不同应用进程
    - 在网络中采用发送方和接收方的套接字（Socket）组合来识别端点
      - 套接字=（主机IP地址，端口号）
      - 唯一地标识网络中的一台主机和其上的一个应用（进程）
      - 实际上是一个通信端点
- 无连接服务与面向连接服务
  - 面向连接服务
    - 在通信双方进行通信之前，必须先建立连接，在通信过程中，整个连接的 情况一直被实时地监控和管理
    - 通信结束后，应该释放这个连接
  - 无连接服务 两个实体之间的通信不需要先建立好连接，需要通信时，直接将信息发送到"网络"中，让该信息的传递在网上尽力而为地往目的地传送
  - TCP/IP协议族的传输协议
    - 面向连接的传输控制协议（TCP），：采用TCP时，传输层向上提供的是一条全双工的可靠逻辑信道
    - 无连接的用户数据报协 议（UDP）：采用UDP时，传输层向上提供的是一条不可靠的逻辑信道
  - TCP/IP协议
    - TCP提供面向连接的可靠传输服务，增加了许多开销，如确认、流量控制、计时器及连接管理等
    - TCP主要适用于可靠性更重要的场合，如文件传输协议（FTP）、 超文本传输协议（HTTP）、远程登录（TELNET）等
  - UDP协议
    - 在IP之上仅提供两个附加服务：多路复用和对 数据的错误检查
    - 远程主机的传输层收到UDP报文后，不需要给出任何确认
    - UDP简单，执行速度比较快、实时性好
    - UDP的应用有：小文件传送协议（TFTP）、DNS、SNMP和实时传输协议（RTP）
- 注意
  - IP数据报和UDP数据报的区别
    - IP数据报在网络层要经过路由的存储转发
    - UDP数据报在传输层的端到端的逻辑信道中传输，封装成IP数据报在网络层传输时，UDP数据报的信息对路由是不可见的
  - TCP和网络层虚电路的区别
    - 一
      - TCP报文段在传输层抽象的逻辑信道中传输，对路由器不可见
      - 虚电路所经过的交换结点都必须保存虚电路状态信息
    - 二
      - 网络层若采用虚电路方式，则无法提供无连接服务
      - 而传输层采用TCP不影响网络层提供无连接服务

5.2 UDP协议

UDP数据报

UDP实现了传输协议能够做的最少工作，在IP的数据报服务之上增 加了两个最基本的服务：复用和分用以及差错检测

- UDP优点
  - UDP无须建立连接　　不会有建立连接的时延
  - 无连接状态　　UDP不维护连接状态，应用服务器使用UDP时，一般都能支持更多的活动客户机
  - 分组首部开销小
- 特点
  - UDP常用于一次性传输较少数据的网络应用.如DNS、SNMP等
  - UDP提供尽最大努力的交付，即不保证可靠交付　　所有维护传输可靠性的工作需要用户在应用层来完成
  - UDP是面向报文的　　报文是UDP数据报处理的最小单位
- UDP的首部格式
  - 源端口：源端口号。在需要对方回信时选用，不需要时可用全0
  - 目的端口：这在终点交付报文时必须使用到。
  - 长度：UDP数据报的长度（包括首部和数据），其最小值是8（仅有首部）
  - 校验和
    - 检测UDP数据报在传输中是否有错。有错就丢弃
    - 该字段是可选的，当源主机 不想计算校验和时，则直接令该字段为全0
- 收到数据报
  - 根据首部中的目的端口， 把UDP数据报通过相应的端口上交给应用进程
  - 如果接收方UDP发现收到的报文中的目的端口号不正确，丢弃该报文，并由ICMP发送“端口不可达”差错报文给发送方

UDP 校验

UDP的校验和只检查首部和数据部分

- 注意
  - 校验时，若UDP数据报部分的长度不是偶数个字节，则需填入一个全0字节
  - 如果UDP校验和校验出UDP数据报是错误的，那么可以丢弃，也可以交付给上层，但是需要附上错误报告
  - 通过伪首部，不仅可以检查源端口号、目的端口号和UDP用户数据报的数据部分，还可 以检查IP数据报的源IP地址和目的地址

检错能力并不强

优点：简单、处理速度快

5.3 TCP协议（上）

- **TCP协议的特点**
  - TCP是在不可靠的IP层之上实现的可靠的数据传输协议，它主要解决传输的可靠、有序、 无丢失和不重复问题
  - 特点
    - TCP是面向连接的传输层协议
    - 每条TCP连接只能有两个端点，每条TCP连接只能是点对点的（一对一）
    - TCP提供可靠的交付服务，保证传送的数据无差错、不丢失、不重复且有序
    - TCP提供全双工通信，允许通信双方的应用进程在任何时候都能发送数据，为此TCP连 接的两端都设有发送缓存和接收缓存，用来临时存放双向通信的数据
    - TCP是面向字节流的
- **TCP报文段**
  - TCP传送的数据单元称为报文段
  - 作用
    - 运载数据
    - 建立连接、释放连接和应答
  - 字段意义
    - 源端口和目的端口字段：各占2B，端口是运输层与应用层的服务接口，运输层的复用和分用功能都要通过端口实现
    - 序号字段
      - 占4B，TCP是面向字节流的（即TCP传送时是逐个字节传送的），所以TCP 连接传送的数据流中的每个字节都编上一个序号
      - 序号字段的值指的是本报文段所发送的数据的第一个字节的序号
    - 确认号字段：占4B,是期望收到对方的下一个报文段的数据的第一个字节的序号
    - 数据偏移（即首部长度）：占4位，它指出TCP报文段的数据起始处距离TCP报文段的起始处有多远
    - 保留字段：占6位，保留为今后使用
    - 紧急位URG：URG= 1时，表明紧急指针字段有效。它告诉系统报文段中有紧急数据, 应尽快传送（相当于高优先级的数据）
    - 确认位ACK
      - ACK=1时确认号字段才有效
      - ACK = 0时，确认号无效

      tcp规定，在连接建立后所有传送的报文段都必须把ack置1
    - 推送位PSH （Push）：接收TCP收到PSH=1的报文段，就尽快地交付给接收应用进程， 而不再等到整个缓存都填满后再向上交付
    - 复位位RST （Reset）：RST=1时，表明TCP连接中出现严重差错（如主机崩溃或其他原因），必须释放连接，然后再重新建立运输连接
    - 同步位SYN：SYN=1表示这是一个连接请求或连接接收报文
    - 终止位FIN （Finish）:用来释放一个连接。FIN= 1表明此报文段的发送方的数据已发送完毕，并要求释放传输连接
    - 窗口字段:占2B，表示允许对方发送的数据量，单位为字节
    - 校验和：占2B，校验和字段检验的范围包括首部和数据两部分
    - 紧急指针字段：占16位，指出在本报文段中紧急数据共有多少字节（紧急数据放在本报文段数据的最前面）
    - 选项字段：长度可变，TCP最初只规定了一种选项，即最大报文段长度
    - 填充字段：使整个首部长度是4B的整数倍

5.3 TCP协议（中）

TCP连接管理

TCP连接的管理就是使运输连接的建立和释放都能正常进行

连接阶段
- 连接建立
- 数据传送
- 连接释放

TCP连接建立时面对的问题
- 要使每一方都能够确知对方的存在
- 要允许双方协商一些参数（如最大窗口值、是否使用窗口扩大选项、时间戳选项及服务质量等）
- 能够对运输实体资源（如缓存大小、连接表中的项目等）进行分配

连接的建立(三次握手）
- 客户机的TCP首先向服务器的TCP发送一个连接请求报文段
  - SYN=1
  - seq=x
- 服务器的TCP收到连接请求报文段后，如同意建立连接，就向客户机发回确认，并为该TCP连接分配TCP缓存和变量
  - SYN = 1
  - ACK = 1
  - ack= x+1
  - seq = y
- 当客户机收到确认报文段后，还要向服务器给出确认，并且也要给该连接分配缓存和变量
  - ACK = 1
  - seq = x+1
  - ack = y+1

服务器易于受到syn洪泛攻击

连接的释放（四次挥手）
- 客户机向其TCP发送一个连接释放报文段，并停止发送数据，主动关闭TCP连接
  - FIN = 1
  - seq = u
- 服务器收到连接释放报文段后即发出确认
  - ACK =1
  - seq = v
  - ack = u+1
- 服务器通知客户端TCP释放连接
  - FIN = 1
  - ACK = 1
  - seq = w
  - ack = u+1
- 客户机受到连接释放报文后，发出确认
  - ACK = 1
  - seq = u+1
  - ack = w+1

SYN洪泛攻击

SYN洪泛攻击发生在OSI第四层，这种方式利用TCP协议的特性，就是三次握手

攻击者发送TCP SYN，SYN是TCP三次握手中的第一个数据包，而当服务器返回ACK后，该攻击者就不对其进行再确认

那这个TCP连接就处于挂起状态，也就是所谓的半连接状态，服务器收不到再确认的话，还会重复发送ACK给攻击者

影响
- 浪费服务器的资源
- 在服务器上，这些TCP连接会因为挂起状态而消耗CPU和内存，最后服务器可能死机

5.3 TCP协议（下）

- **TCP可靠传输（实现机制）**
  - 序号：TCP首部的序号字段用来保证数据能有序提交给应用层，序号建立在传送的字节流之上
  - 确认
    - TCP首部的确认号是期望收到对方的下一个报文段的数据的第一个字节的序号
    - TCP默认使用累计确认，即TCP只确认数据流中至第一个丢失字节为止的字节
  - 重传
    - 超时：TCP每发送一个报文段，就对这个报文段设置一次计时器。计时器设置的重传时间到期但还未收到确认时，就要重传这一报文段
    - 冗余ACK
      - 再次确认某个报文段的 ACK,而发送方先前已经收到过该报文段的确认
      - 当收到对于某个报文段的3个冗余ACK，可以认为该报文段已经丢失。这时发送方可以立即对该报文执行重传
- **TCP流量控制**
  - 匹配发送方的发送速率与接收方的读取速率
  - 流量控制机制
    - 基于滑动窗口协议的流量控制机制
    - 实现方法
      - 接收方根据自己接收缓存的大小，动态地调整发送方的发送窗口大小（接收窗口 rwnd），限制发送方向网络注入报文的速率
      - 发送方根据其对当前网络拥塞程序的估计而确定的窗口值，这称为拥塞窗口 cwnd　　其大小与网络的带宽和时延有关
  - 流量控制的区别
    - 传输层：定义端到端用户之间的流量控制
    - 数据链路层：定义两个中间的相邻结点的流量控制
  - 窗口大小的区别
    - 传输层：滑动窗口可以动态变化
    - 数据链路层：滑动窗口不能动态变化
- **TCP拥塞控制**
  - 拥塞控制：防止过多的数据注入网络，保证网络中的路由器或链路不致过载
  - 拥塞控制与流量控制的区别
    - 相同点：它们都通过控制发送方发送数据的速率来达到控制效果
    - 区别
      - 拥塞控制是让网络能够承受现有的网络负荷，是一个全局性的过程，涉及所有的主机、所有的路由器，以及与降低网络传输性能有关的所有因素
      - 流量控制是点对点的通信量的控制，即接收端控制发送端，它所要做的是抑制发送端发送数据的速率，以便使接收端来得及接收
  - 窗口
    - 接收窗口 rwnd：接收方根据目前接收缓存大小所许诺的最新窗口值，反映接收方的容量
    - 拥塞窗口 cwnd：发送方根据自己估算的网络拥塞程度而设置的窗口值，反映网络的当前容量
    - ***发送窗口的上限值=min[rwnd, cwnd]***
  - 实现机制
    - 慢开始
      - 每经过一个传输轮次（即往返时延RTT）,拥塞窗口 cwnd就会加倍
      - 令拥塞窗口 cwnd = 1即一个最大报文段长度MSS，每收到一个对新报文段的确认后，将cwnd加1
      - 使分组注入网络的速率更加合理
      - 慢开始一直把拥塞窗口 cwnd增大到一个规定的慢开始门限ssthresh （阈值），然后改用拥塞避免算法
    - 拥塞避免
      - 发送端的拥塞窗口 cwnd每经过一个往返时延RTT就增加一个MSS 的大小，而不是加倍
      - cwnd按线性规律缓慢增长（即加法增大），而当出现一次超时（网络拥塞） 时，令慢开始门限ssthresh等于当前cwnd的一半（即乘法减小）
    - 快重传：当发送方连续收到三个重复的ACK报文时，直接重传对方尚未收到的报文段，而不必等待那个报文段设置的重传计时器超时
    - 快恢复
      - 发送端收到连续三个冗余ACK （即重复确认）时，执行"乘法减小"算法，把慢开始门限ssthresh设置为出现拥塞时发送方cwnd的一半
      - 把cwnd的值设置为慢开始门限ssthresh改变后的数值， 然后开始执行拥塞避免算法（ “加法增大” ）,使拥塞窗口缓慢地线性增大
  - 网络拥塞的处理
    - 发送方检测到超时事件的发生（未按时收到确认，重传计时器超时），就要把慢开始门限ssthresh设置为出现拥塞时的发送方的cwnd值的一半（但不能小于2）
    - 然后把拥塞窗口 cwnd重新设置为L执行慢开始算法